A

11/19/99 C7132 U.S. PTO

Please type a plus sign (+) inside this box → [+]

PTO/SB/05 (4/98)
Approved for use through 09/30/2000. OMB 0651-0032
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 C.F.R. § 1.53(b))

Attorney Docket No.	130.1012.02
First Inventor or Application Identifier	David L. Isaman
Title	Symbolic Store-Load By Pass
Express Mail Label No.	EK 025 321 010 US

jc590 U.S. PTO 09/443160 11/19/99

APPLICATION ELEMENTS
See MPEP chapter 600 concerning utility patent application contents.

ADDRESS TO: **Assistant Commissioner for Patents**
Box Patent Application
Washington, DC 20231

1. [ ] * Fee Transmittal Form (e.g., PTO/SB/17)
   (Submit an original and a duplicate for fee processing)
2. [X] Specification [Total Pages 20]
   (preferred arrangement set forth below)
   - Descriptive title of the Invention
   - Cross References to Related Applications
   - Statement Regarding Fed sponsored R & D
   - Reference to Microfiche Appendix
   - Background of the Invention
   - Brief Summary of the Invention
   - Brief Description of the Drawings (if filed)
   - Detailed Description
   - Claim(s)
   - Abstract of the Disclosure
3. [X] Drawing(s) (35 U.S.C. 113) [Total Sheets 3]
4. Oath or Declaration [Total Pages]
   a. [ ] Newly executed (original or copy)
   b. [ ] Copy from a prior application (37 C.F.R. § 1.63(d))
      (for continuation/divisional with Box 16 completed)
      i. [ ] DELETION OF INVENTOR(S)
         Signed statement attached deleting inventor(s) named in the prior application, see 37 C.F.R. §§ 1.63(d)(2) and 1.33(b).

NOTE FOR ITEMS 1 & 13: IN ORDER TO BE ENTITLED TO PAY SMALL ENTITY FEES, A SMALL ENTITY STATEMENT IS REQUIRED (37 C.F.R. § 1.27), EXCEPT IF ONE FILED IN A PRIOR APPLICATION IS RELIED UPON (37 C.F.R. § 1.28).

5. [ ] Microfiche Computer Program (Appendix)
6. Nucleotide and/or Amino Acid Sequence Submission
   (if applicable, all necessary)
   a. [ ] Computer Readable Copy
   b. [ ] Paper Copy (identical to computer copy)
   c. [ ] Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

7. [ ] Assignment Papers (cover sheet & document(s))
8. [ ] 37 C.F.R.§3.73(b) Statement (when there is an assignee) [ ] Power of Attorney
9. [ ] English Translation Document (if applicable)
10. [ ] Information Disclosure Statement (IDS)/PTO-1449 [ ] Copies of IDS Citations
11. [ ] Preliminary Amendment
12. [X] Return Receipt Postcard (MPEP 503)
    (Should be specifically itemized)
13. [ ] * Small Entity Statement(s) (PTO/SB/09-12) [ ] Statement filed in prior application, Status still proper and desired
14. [ ] Certified Copy of Priority Document(s)
    (if foreign priority is claimed)
15. [X] Other: Return Receipt Postcard; certificate of mailing

16. **If a CONTINUING APPLICATION,** check appropriate box, and supply the requisite information below and in a preliminary amendment:
[ ] Continuation [ ] Divisional [ ] Continuation-in-part (CIP) of prior application No: ______/______
Prior application information: Examiner ______ Group / Art Unit: ______
For CONTINUATION or DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 4b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

17. CORRESPONDENCE ADDRESS

[ ] Customer Number or Bar Code Label (Insert Customer No. or Attach bar code label here) or [X] Correspondence address below

Name	Lisa K. Jorgenson				
	STMicroelectronics, Inc.				
Address	1310 Electronics Drive				
	MS 2346				
City	Carrolton	State	TX	Zip Code	75006
Country	US	Telephone	(972) 466-7417	Fax	(972) 466-7044

Name (Print/Type)	Steven A. Swernofsky	Registration No. (Attorney/Agent)	33,040
Signature	S A Swernofsky	Date	11-18-99

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231.

Certificate of Mailing (37 C.F.R. § 1.10)

I hereby certify that this paper (along with any paper referred to as being attached or enclosed) is being deposited with the United States Postal Services on the date shown below as "Express Mail" (Post Office to Addressee) in an envelope addressed to the Commissioner of Patents and Trademarks, Washington, D.C. 20231.

Mailing Label No. **EK 025 321 010 US**

Date of Deposit: November 19, 1999

Arlette Malhas
Printed Name

Arlette Malhas
Signature

Documents enclosed:

- Utility Patent Application Transmittal Form (SB/05); Title Page; Application (Specification, 15 pgs.; Claims, 1 pg.; Abstract,1 pg.; Drawings 3 pgs.); and Certificate of Express Mail Mailing

This application is submitted in the name of the following inventor(s):

Inventor	Citizenship	Residence City and State
David L. Isaman	United States	San Diego, California

The assignee is MetaFlow Technology, Inc., having an office at 4250 Executive Square, Suite 300, La Jolla, CA 92037.

Title of Invention

Symbolic Store-Load Bypass

Related Applications

This application claims priority to copending provisional application number 06/114,295 entitled "Symbolic Store-Load Bypass", filed December 31, 1998, by the same inventor.

The inventions described herein can be used in combination or conjunction with inventions described in the following patent applications (2):

- Application Serial No. 60/114296, Express Mail Mailing No. EE506030698US, filed December 31, 1998, in the name of Anatoly Gelman, titled "Call Return Branch Production Buffer," assigned to the same assignee, attorney docket number META-013, and all pending cases claiming priority thereof; and

- Application Serial No. 06/14,297, Express Mail Mailing No. EE506030684US, filed December 31, 1998, in the name of Anatoly Gelman and Russell Schapp, titled "Block-Based Branch Table Buffer," assigned to the same assignee, attorney docket number META-014, and all pending cases claiming priority thereof.

These applications are hereby incorporated by reference as if fully set forth herein. These applications are collective referred to herein as "incorporated disclosures".

Background of the Invention

1. Field of the Invention

This invention relates to microprocessor design.

2. *Related Art*

In microprocessors employing pipelined architecture, it is desirable to be in the process of executing as many instructions as possible, so that each element of the pipeline is maintained busy. However, some instructions, such as instructions that load data from external memory or stage data into external memory, must generally be executed in their original sequence order, so as to avoid the external memory ever being in an incorrect state. Moreover, when such instructions refer to identical external memory locations, where is no particular need to wait for the actual external memory operations to complete, as the identical data is already available for the processor to operate with.

One problem in the known art is that determining whether two different instructions refer to the identical location in external memory generally requires computing the actual external memory address referenced by each of the two different instructions. This prolongs when the determination can be made, because it requires time (and typically, a pipeline stage) to actually compute the referenced external memory addresses.

Accordingly, it would be advantageous to provide a technique for operating a pipelined microprocessor more quickly, by detecting instructions that load from identical memory locations as were recently stored to, without having to actually compute the referenced external memory addresses. In a preferred embodiment, the microprocessor examines the symbolic structure of instructions as they are encountered, so as to be able

to detect identical memory locations by examination of their symbolic structure. For example, instructions that store to and load from an identical offset from an identical register are determined to be referencing the identical memory locations, without having to actually compute the complete physical target address.

Summary of the Invention

The invention provides a method and system for operating a pipelined microprocessor more quickly, by detecting instructions that load from identical memory locations as were recently stored to, without having to actually compute the referenced external memory addresses. The microprocessor examines the symbolic structure of instructions as they are encountered, so as to be able to detect identical memory locations by examination of their symbolic structure. For example, in a preferred embodiment, instructions that store to and load from an identical offset from an identical register are determined to be referencing the identical memory location, without having to actually compute the complete physical target address.

///

///

///

Brief Description of the Drawings

Figure 1 shows a block diagram of a system in a pipelined microprocessor for detecting identical locations referenced by different load and store instructions.

Figure 2 shows a process flow diagram of a method for operating a system in a pipelined microprocessor for detecting identical locations referenced by different load and store instructions.

Detailed Description of the Preferred Embodiment

In the following description, a preferred embodiment of the invention is described with regard to preferred process steps and data structures. Embodiments of the invention can be implemented using circuits in a microprocessor or other device, adapted to particular process steps and data structures described herein. Implementation of the process steps and data structures described herein would not require undue experimentation or further invention.

///

///

///

///

System Elements

Figure 1 shows a block diagram in a pipelined microprocessor for detecting identical locations referenced by different load and store instructions.

A microprocessor 100 includes a sequence of pipeline stages, including an instruction fetch state 110, an instruction decode state 120, an address computation state 130 and an instruction execution state 140. In a preferred embodiment, the pipeline stages of the microprocessor 100 operate concurrently on sequences of instructions 151 in a pipelined manner. Pipeline operation is known in the art of microprocessor design.

In operation, the microprocessor 100 is coupled to an instruction memory 150 which includes a plurality of instructions 151, at least some of which are memory load or store instructions. In a preferred embodiment, the instruction memory 150 includes a random access memory. Memory caching operations can be performed either by the instruction memory 150, input and output elements of the microprocessor 100, or both. Memory caching operations, as well as other aspects of reading and writing memory locations, are known in the art of computer memories and so are not further described herein.

The microprocessor 100 reads a sequence of instructions 151 from the instruction memory 150 using the instruction fetch stage 110 (and including any associated

memory read or write elements in the microprocessor 100). In a preferred embodiment, the input instruction buffer 110 includes a plurality of instructions 151 from the instruction memory 150, but there is no particular requirement therefore.

The instruction fetch stage 110 couples the instructions to the instruction decode state 120.

The instruction decode stage 120 parses the instructions 151 to determine what types of instructions 151 they are (such as instructions 151 that load data from external memory or store data to external memory). As part of the parsing instructions 151, and in addition to determine what operations the instructions 151 command the microprocessor 100 to perform, the instruction decode stage 120 determines the syntax of any addresses in the external memory that the instructions 151 refer to as operands.

For example, an instruction that loads data from external memory has a format that refers to the specific location in external memory from which to load the data. The format can include a base address value and an offset address value, which are to be added to compute the effective reference address of the instruction 151. The base address value can be a constant value or specify a value found in an internal register of the microprocessor 100. Similarly, the offset address value can be a constant value or specify a value found in an internal register of the microprocessor.

Similarly, an instruction that stores data to external memory has a format that refers to the specific location in external memory from which to store the data. The format can similarly include a base address value and an offset address value, which are used to compute the effective reference address of the instruction 151.

The instruction decode stage 120 couples the parts of the instruction 151, including information about the base address value and the offset address value, to the address computation stage 130.

The address computation stage 130 receives the base address value and the offset address value, and computes the effective reference address of the instruction 151.

The instruction decode stage 120 couples the parts of the instruction 151, including information about what operations the instructions 151 command the micro-processor 100 to perform, and what the syntax of any addresses the instructions 151 refer to as operands, to the instruction execution stage 140. The address computation stage 130 couples the effective reference address of the instruction 151, to the instruction execution stage 140.

The instruction decode stage 120 includes a symbolic load-store bypass element 121. The bypass element 121 examines the parts of the instruction 151, including information about what operations the instructions 151 command the microprocessor

100 to perform. If these operations are to load data from external memory, or to store data to external memory, the bypass element 121 further examines the syntax of any addresses 151 refer to as operands.

If the operand addresses the instructions 151 refer to include identical base address values and offset address values, the bypass element 121 generates a bypass signal indicating that the instructions 151 refer to the same location in external memory.

When the bypass signal is generating, the address computation stage 130, does not have to compute the actual effective address for the microprocessor 100 to act on the knowledge that the instructions 151 refer to identical locations in external memory.

For example, suppose that a first instruction 151 to store data refers to a location in external memory determined as (contents of register A) + (fixed offset value B), and a second instruction 151 to load data refers to the same location in external memory determined as (contents of register A) + (fixed offset value B), where A and B are identical. In this case, the microprocessor 100 can proceed with the knowledge that the first (store) instruction 151 and the second (load instruction) 151 refer to the same location. Since the second (load) instruction 151 is going to read the same data from external memory that the first (store) instruction 151 put there, the microprocessor 100 can proceed by using that data from an internal register, rather than waiting for external memory to complete actual store and load operations.

Although the actual first (store) instruction 151 would be physically performed and completed by external memory, the microprocessor 100 can proceed without physically performing the second (load) instruction 151. Instead, the microprocessor 100 can use the identical data from it's internal register, thus removing a relative delay in microprocessor 100 operation.

Method of Operation

Figure 2 shows a process flow diagram of a method for operating a system in a pipelined microprocessor for detecting identical locations referenced by different load and store instructions.

A method 200 is performed by the microprocessor 100, including its sequence of pipeline stages. In a preferred embodiment, as many steps of the method 200 are performed concurrently in a pipelined manner. Pipeline operation is known in the art of microprocessor design.

At a flow point 210, microprocessor 100 is coupled to an instruction memory 150, which includes a plurality of instructions 151, and is ready to perform those instructions 151. At least some of those instructions 151 are memory load or store instructions.

At a flow point 211, the microprocessor reads a sequence of instructions 151 from the memory 150 using the instruction fetch stage 110 (and including any associated memory read or write elements in the microprocessor 100).

At a step 212, the instruction fetch stage 110 couples the instructions 151 to the instruction decode stage 120.

At a step 213(a), the instruction decode stage 120 parses the instructions 151 to determine whether they are instructions 151 that load data from external memory or store data to external memory.

At a step 213(b), the instruction decode stage 120 determines the syntax of any addresses in the external memory that the instructions 151 refer to as operands.

At a step 214, the bypass element 121 examines the parts of the instruction 151, including information about what operations the instructions 151 command the microprocessor 100 to perform. If these operations are to load data from external memory, or to store data to external memory, the method continues with the step 215. If these operations are otherwise, the method continues with the step 221.

In a step 215, a record of the symbolic operands of the store operations to external memory is stored in a table that is indexed by the instruction ID.

In a step 216, each load instruction's operands are compared against both the store instructions being issued in the ongoing clock cycle and those of all unretired store instructions. By storing the record of these operations for comparison, there is a much higher probability of detecting a useful bypass in subsequent steps where the bypass element 121 further examines the syntax of any addresses the instructions 151 refer to as operands.

At a step 217, the bypass element 121 determines whether the operand addresses that the instructions 151 refer to include identical base address values and offset address values. If so, the bypass element 121 generates a bypass signal indicating that the instructions 151 refer to the same location in external memory. If not, the bypass element 121 does not generate a bypass signal. (In alternative embodiments, the bypass element 121 may generate an inverse bypass signal). If the bypass element 121 generates a bypass signal, the method 200 proceeds with the step 216. If not, the method 200 proceeds with the step 221.

At a flow point 220, the bypass signal having been generated, the microprocessor 100 can act on the knowledge that the instructions 151 refer to identical locations in external memory. For example, if a first (store) instruction 151 and a second (load) instruction 151 refer to identical locations in external memory, the microprocessor 100 can proceed by using data to be transferred by those instructions 151 from an internal

register. The microprocessor 100 does not have to wait for external memory to complete actual store and load operations.

At a step 221, the instruction decode stage 120 couples the parts of the instruction 151, including information about the base address value and the offset address value to the address computation stage 130.

At a step 222, the address computation stage 130 receives the base address value and the offset address value, and computes the effective reference address of the instruction 151.

At a step 223, the instruction decode stage 120 couples the parts of the instruction 151, including information about what operations the instructions 151 command the microprocessor 100 to perform, and what the syntax of any address the instructions 151 refer to as operands, to the instruction execution stage 140.

At a step 224, the address computation stage 130 couples the effective reference address of the instruction 151, to the instruction execution stage 140.

At a step 225, the first (store) instruction 151 is physically performed and completed by external memory.

At a step 226(a), if the bypass signal was generated, the microprocessor 100 proceeds without physically performing the second (load) instruction 151. Instead, the microprocessor 100 can use the identical data from it's internal register, thus removing a relative delay in microprocessor 100 operation.

Alternatively, at a step 226(b), if the bypass signal was not generated, or in if an inverse bypass signal was generated, second (load) instruction 151 is physically performed and completed by external memory.

Alternative Embodiment

Although preferred embodiments are disclosed herein, many variations are possible which remain within the concept, scope and spirit of the invention, and these variations would become clear to those skilled in the art after perusal of this application.

Claims

1. A method for operating a pipelined microprocessor, said method including steps for

detecting a first instruction that stores to a first memory location, said first instruction including syntax for computing an effective address for said first memory location;

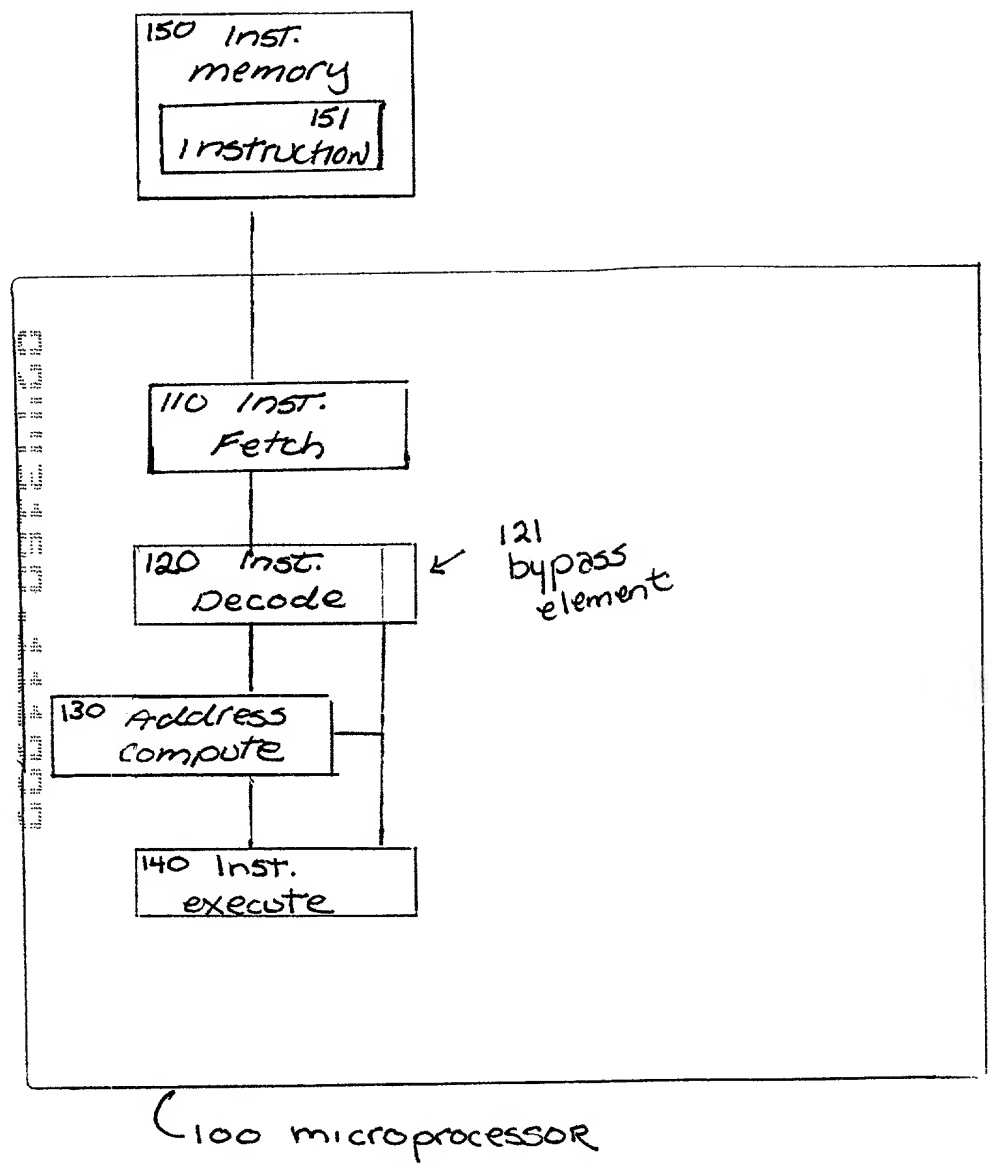
detecting a second instruction that stores to a second memory location, said second instruction including syntax for computing an effective address for said second memory location;

determining, in response to said syntax for said first instruction and said syntax for said second instruction, a relationship between said first memory location and said second memory location, without computing said effective address for both said first memory location and said second memory location; and

determining whether to perform one of said first instruction and second instruction in response to said step of determining a relationship.

Abstract

The invention provides a method and system for operating a pipelined microprocessor more quickly, by detecting instructions that load from identical memory locations as were recently stored to, without having to actually compute the referenced external memory addresses. The microprocessor examines the symbolic structure of instructions as they are encountered, so as to be able to detect identical memory locations by examination of their symbolic structure. For example, in a preferred embodiment, instructions that store to and load from an identical offset from an identical register are determined to be referencing the identical memory location, without having to actually compute the complete physical target address.


150 Inst. memory
151 Instruction
110 Inst. Fetch
120 Inst. Decode
121 bypass element
130 Address compute
140 Inst. execute
100 microprocessor

META-012
Fig. 2A

210
Microprocessor is coupled to an instruction memory 150 which includes a plurality of instructions 151, and is ready to perform those instructions 151.

211
The microprocessor reads a sequence of instructions 151 from the memory 150 using instruction fetch stage 110.

212
Instruction fetch stage 110 couples the instructions 151 to the instruction decode stage 120.

213(a)
The instruction decode stage 120 parses the instructions 151 to determine whether they are instructions to load data to an external memory or store data from an external memory.

213(b)
The instruction decode stages 120 determines the syntax of any addresses in the external memory that the instructions 151 refer to as operands.

214
The bypass element 121 examines parts of the instruction 151, including information about what operations the instructions 151 command the microprocessor 100 to perform. If these operations are to load or store data, the method continues with step 115. If these operations are otherwise, the method continues with step 221.

215
A record of the symbolic operands of the store operations to external memory is stored in a table that is indexed by the instruction ID.

216
Each load instructions' operands are compared against the store instructions being issued in the ongoing clock cycle and those of all unretired store instructions.

217
The bypass element 121 determines whether the operand addresses that the instructions 151 refer to include identical base address values. If so, the bypass element generates a bypass signal. If not, the bypass element does not generate a bypass signal.

220
The microprocessor can act on the knowledge that the instructions 151 refer to identical locations in an external memory.

221
The instruction decode stage 120 couples the parts of the instruction 151, including information about the base address value and the offset address value to the address computation stage.

TO FIG 2B

META-012
Fig. 2b